

Отчёта по лабораторной работе 8

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений**

Иовлев Максим Андреевич НПИбд-01-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	23
	Список литературы	24

Список иллюстраций

4.1	Файл lab8-1.asm:	9
4.2	Программа lab8-1.asm:	10
4.3	Файл lab8-1.asm:	11
4.4	Программа lab8-1.asm:	12
4.5	Файл lab8-1.asm	13
4.6	Программа lab8-1.asm	14
4.7	Файл lab8-2.asm	15
4.8	Программа lab8-2.asm	15
4.9	Файл листинга lab8-2	16
4.10	ошибка трансляции lab8-2	17
4.11	файл листинга с ошибкой lab8-2	18
4.12	Файл lab8-3.asm	19
4.13	Программа lab8-3.asm	19
4.14	Файл lab8-4.asm	21
4.15	Программа lab8-4.asm	22

Список таблиц

1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

1. Изучите примеры программ.
2. Изучите файл листинга.
3. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c . Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу
4. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 8.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений X и a из 8.6.

3 Теоретическое введение

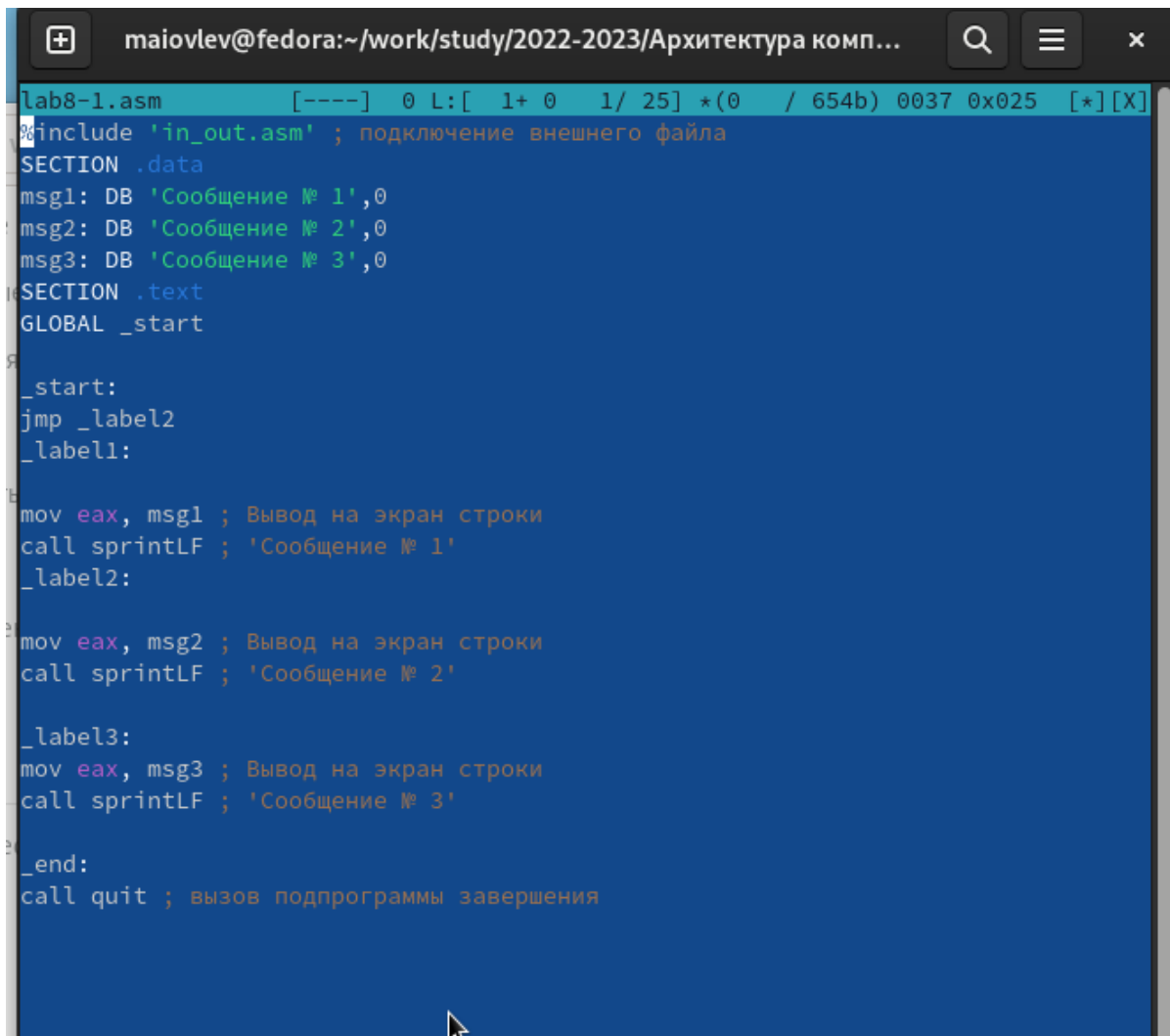
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

4 Выполнение лабораторной работы

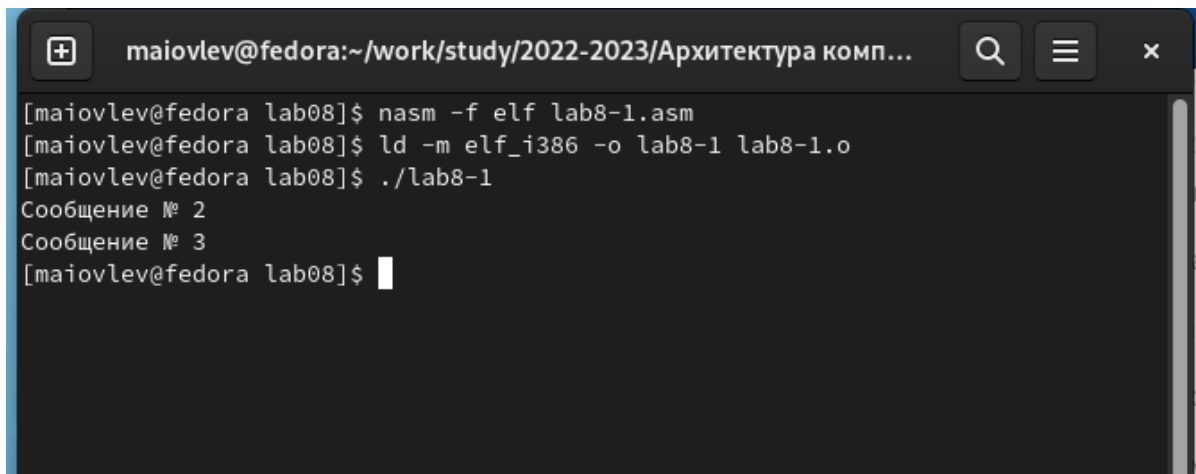
1. Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл `lab8-1.asm`
2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введите в файл `lab8-1.asm` текст программы из листинга 8.1. (рис. 4.1)



```
lab8-1.asm [----] 0 L: [ 1+ 0 1/ 25] *(0 / 654b) 0037 0x025 [*][X]
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.1: Файл lab8-1.asm:

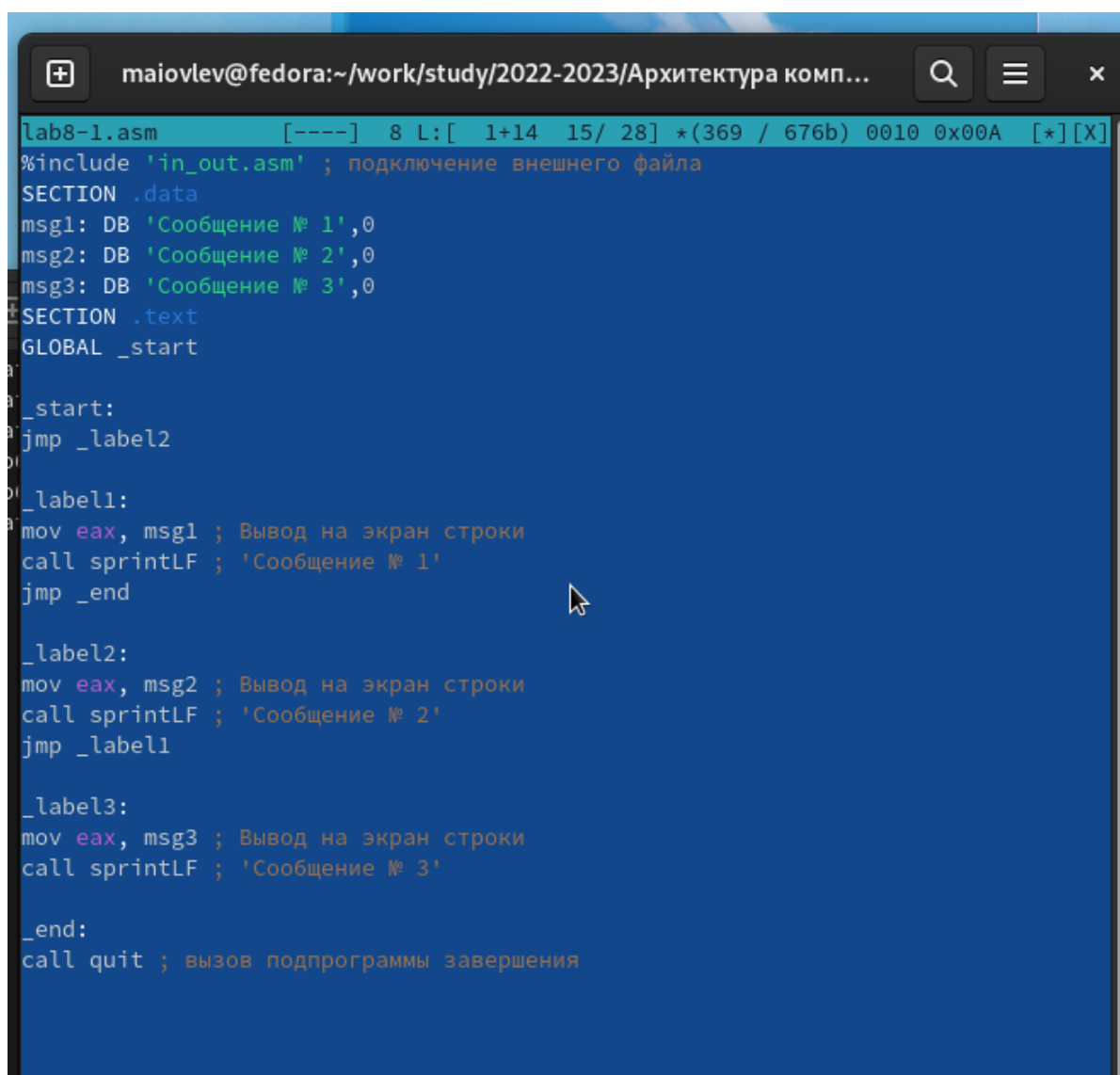
Создайте исполняемый файл и запустите его. (рис. 4.2)

A screenshot of a terminal window with a dark background. The window title is "maiovlev@fedora:~/work/study/2022-2023/Архитектура комп...". The terminal shows the following commands and output:

```
[maiovlev@fedora lab08]$ nasm -f elf lab8-1.asm
[maiovlev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[maiovlev@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[maiovlev@fedora lab08]$
```

Рис. 4.2: Программа lab8-1.asm:

Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Измените текст программы в соответствии с листингом 8.2. (рис. 4.3, 4.4)



```
lab8-1.asm [----] 8 L: [ 1+14 15/ 28] *(369 / 676b) 0010 0x00A [*][X]
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

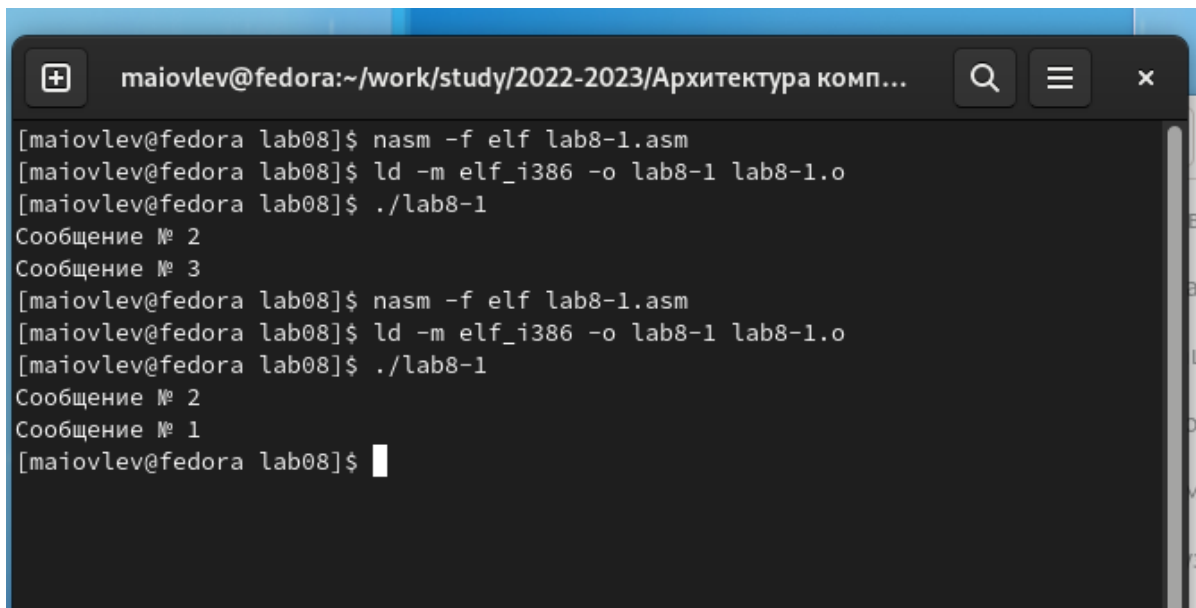
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.3: Файл lab8-1.asm:

A terminal window with a dark background and light text. The window title is "maiovlev@fedora:~/work/study/2022-2023/Архитектура комп...". The terminal shows the following commands and output:

```
[maiovlev@fedora lab08]$ nasm -f elf lab8-1.asm
[maiovlev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[maiovlev@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[maiovlev@fedora lab08]$ nasm -f elf lab8-1.asm
[maiovlev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[maiovlev@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[maiovlev@fedora lab08]$
```

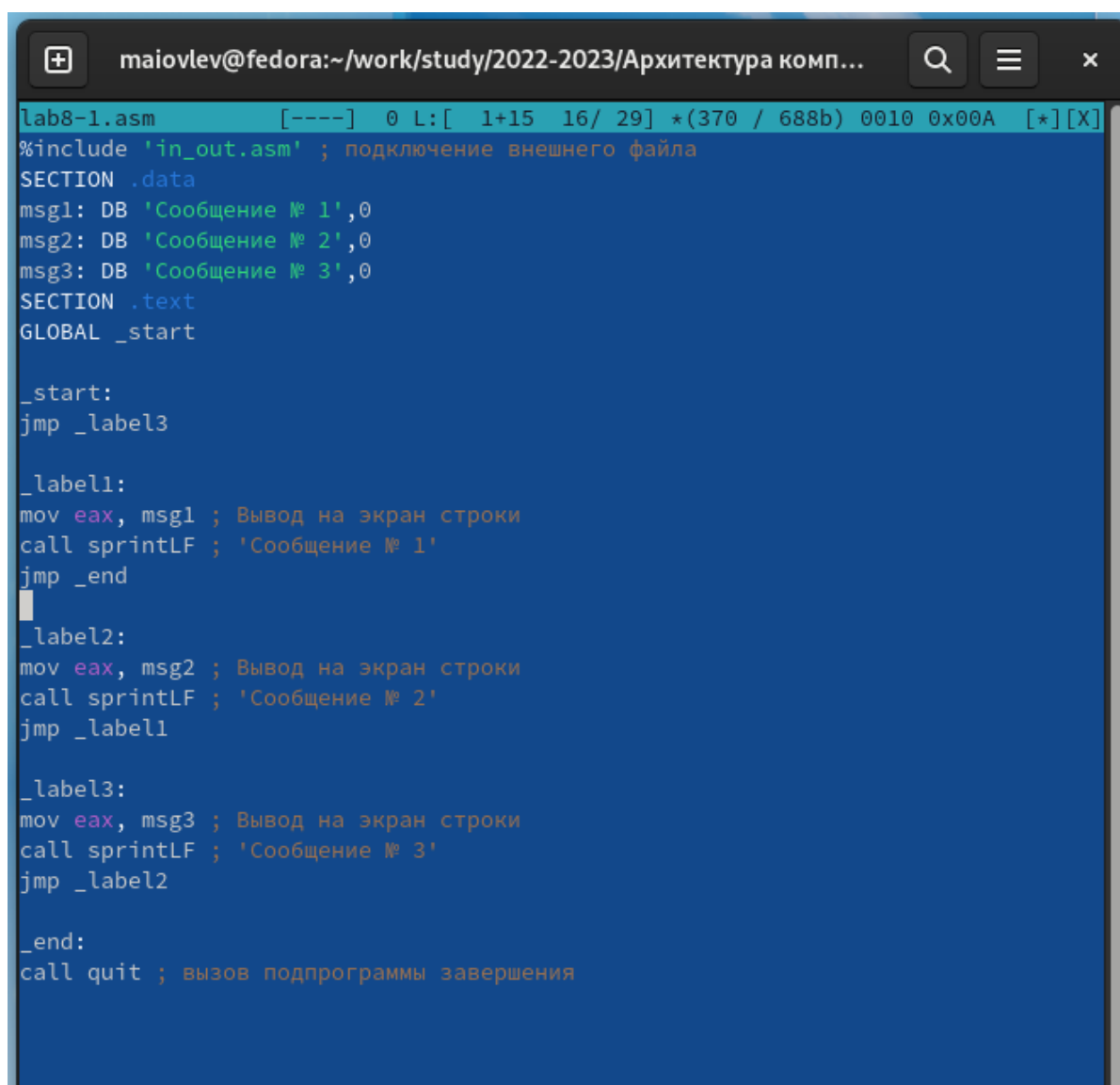
Рис. 4.4: Программа lab8-1.asm:

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. 4.5, 4.6):

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
lab8-1.asm [----] 0 L:[ 1+15 16/ 29] *(370 / 688b) 0010 0x00A [*][X]
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label3

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.5: Файл lab8-1.asm

```
maiovlev@fedora:~/work/study/2022-2023/Архитектура комп...
[maiovlev@fedora lab08]$ nasm -f elf lab8-1.asm
[maiovlev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[maiovlev@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[maiovlev@fedora lab08]$ nasm -f elf lab8-1.asm
[maiovlev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[maiovlev@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[maiovlev@fedora lab08]$ nasm -f elf lab8-1.asm
[maiovlev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[maiovlev@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[maiovlev@fedora lab08]$
```

Рис. 4.6: Программа lab8-1.asm

3. Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры. Создайте исполняемый файл и проверьте его работу для разных значений В. (рис. 4.7, 4.8)

```
lab8-2.asm [----] 0 L: [ 7+ 0 7/ 50] *(135 /1744b) 0115 0x073 [*] [X]
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
```

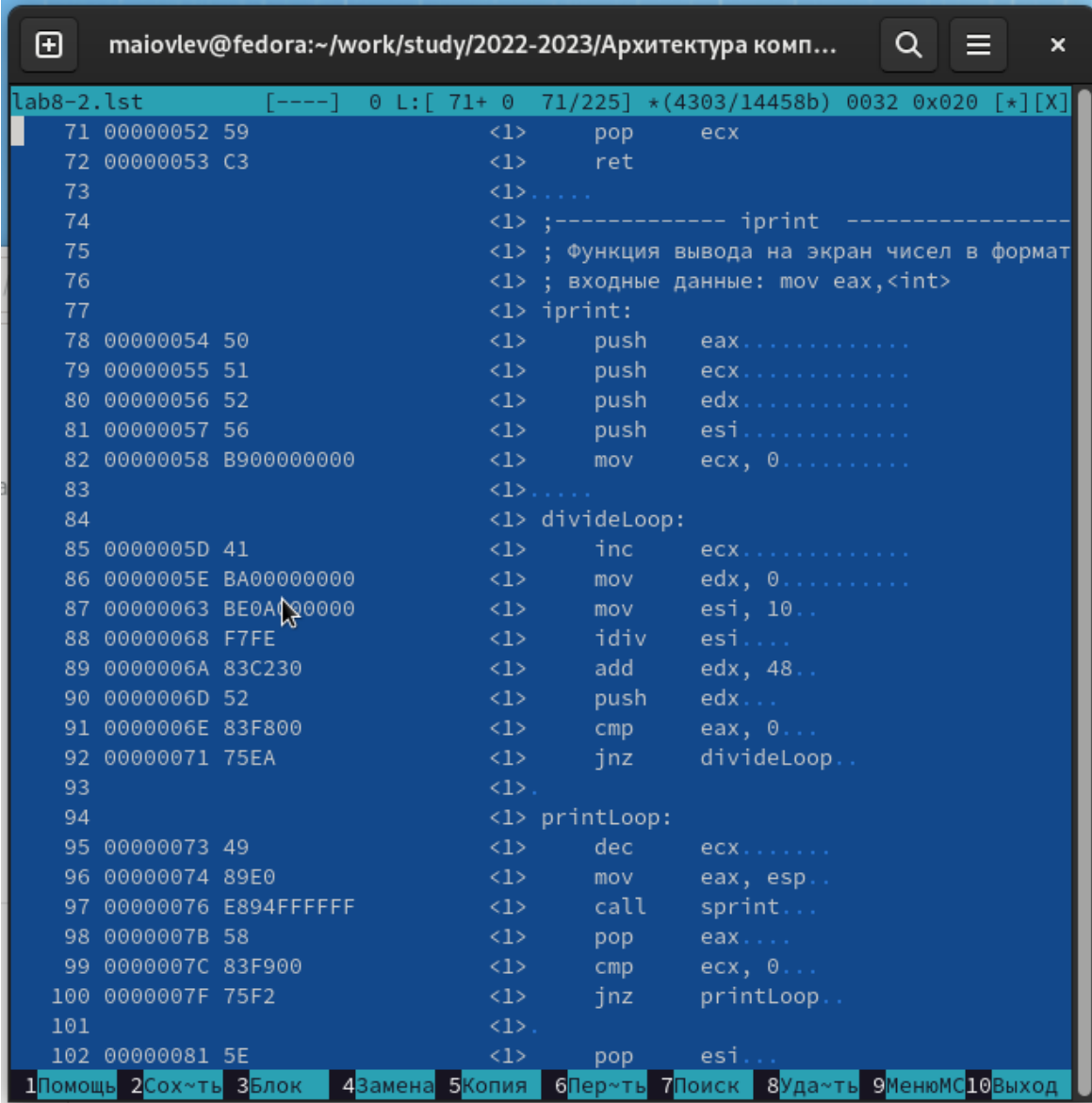
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню 10Выход

Рис. 4.7: Файл lab8-2.asm

```
[maiovlev@fedora lab08]$
[maiovlev@fedora lab08]$ nasm -f elf lab8-2.asm
[maiovlev@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[maiovlev@fedora lab08]$ ./lab8-2
Введите B: 10
Наибольшее число: 50
[maiovlev@fedora lab08]$ ./lab8-2
Введите B: 70
Наибольшее число: 70
[maiovlev@fedora lab08]$
```

Рис. 4.8: Программа lab8-2.asm

4. Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке. Создайте файл листинга для программы из файла `lab8-2.asm` (рис. 4.9)



```
lab8-2.lst [-----] 0 L:[ 71+ 0 71/225] *(4303/14458b) 0032 0x020 [*][X]
71 00000052 59 <1> pop ecx
72 00000053 C3 <1> ret
73 <1>.....
74 <1> ;----- iprint -----
75 <1> ; Функция вывода на экран чисел в формат
76 <1> ; входные данные: mov eax,<int>
77 <1> iprint:
78 00000054 50 <1> push eax.....
79 00000055 51 <1> push ecx.....
80 00000056 52 <1> push edx.....
81 00000057 56 <1> push esi.....
82 00000058 B900000000 <1> mov ecx, 0.....
83 <1>.....
84 <1> divideLoop:
85 0000005D 41 <1> inc ecx.....
86 0000005E BA00000000 <1> mov edx, 0.....
87 00000063 BE0A000000 <1> mov esi, 10..
88 00000068 F7FE <1> idiv esi....
89 0000006A 83C230 <1> add edx, 48..
90 0000006D 52 <1> push edx...
91 0000006E 83F800 <1> cmp eax, 0...
92 00000071 75EA <1> jnz divideLoop..
93 <1>..
94 <1> printLoop:
95 00000073 49 <1> dec ecx.....
96 00000074 89E0 <1> mov eax, esp..
97 00000076 E894FFFFFF <1> call sprint...
98 0000007B 58 <1> pop eax....
99 0000007C 83F900 <1> cmp ecx, 0...
100 0000007F 75F2 <1> jnz printLoop..
101 <1>..
102 00000081 5E <1> pop esi...
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 4.9: Файл листинга `lab8-2`

Внимательно ознакомиться с его форматом и содержанием. Подробно объяснить содержимое трёх строк файла листинга по выбору.

строка 85

- 85 - номер строки
- 0000005D - адрес
- 41 - машинный код
- inc ecx - код программы

строка 86

- 86 - номер строки
- 0000005E - адрес
- BA00000000 - машинный код
- mov edx, 0 - код программы

строка 87

- 87 - номер строки
- 00000063 - адрес
- BE0A000000 - машинный код
- mov esi, 10 - код программы

Откройте файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга (рис. 4.10,4.11)

```
Введите В: 70
Наибольшее число: 70
[maiovlev@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
[maiovlev@fedora lab08]$
[maiovlev@fedora lab08]$
[maiovlev@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:21: error: invalid combination of opcode and operands
[maiovlev@fedora lab08]$
```

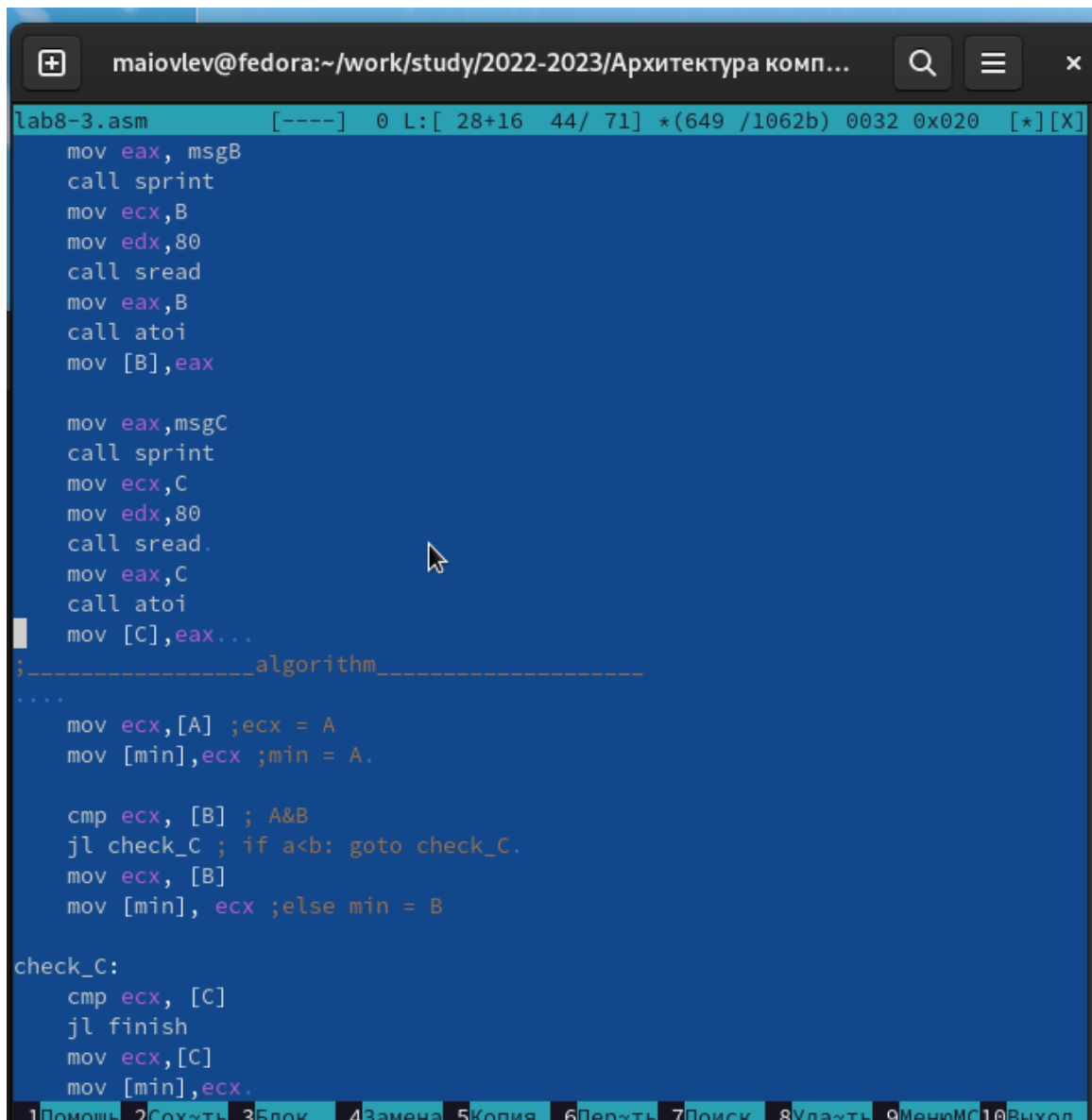
Рис. 4.10: ошибка трансляции lab8-2

```
maiovlev@fedora:~/work/study/2022-2023/Архитектура комп...
lab8-2.lst [----] 57 L:[188+31 219/226] *(14108/14548b) 1076 0x434[*][X]
13 ; ----- Вывод сообщения 'Введите B:
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- преобразование 'B' из симво
21 mov eax,
21 ***** error: invalid combination of opcode an
22 00000101 E896FFFFFF call atoi ; Вызов подпрограммы перевода
23 00000106 A3[00000000] mov [B],eax ; запись преобразованного чи
24 ; ----- Записываем 'A' в переменную
25 0000010B 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
26 00000111 890D[00000000] mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как с
28 00000117 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 0000011D 7F0C jg check_B ; если 'A>C', то переход на м
30 0000011F 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 00000125 890D[00000000] mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' и
33 check_B:
34 0000012B B8[00000000] mov eax,max
35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода
36 00000135 A3[00000000] mov [max],eax ; запись преобразованного
37 ; ----- Сравниваем 'max(A,C)' и 'B'
38 0000013A 8B0D[00000000] mov ecx,[max]
39 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 00000146 7F0C jg fin ; если 'max(A,C)>B', то переход н
41 00000148 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
42 0000014E 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
1Помощь 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```

Рис. 4.11: файл листинга с ошибкой lab8-2

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу (рис. 4.12,4.13)

для варианта 6 - 79,83,41



```
lab8-3.asm [----] 0 L: [ 28+16 44/ 71] *(649 /1062b) 0032 0x020 [*][X]
mov eax, msgB
call sprint
mov ecx, B
mov edx, 80
call sread
mov eax, B
call atoi
mov [B], eax

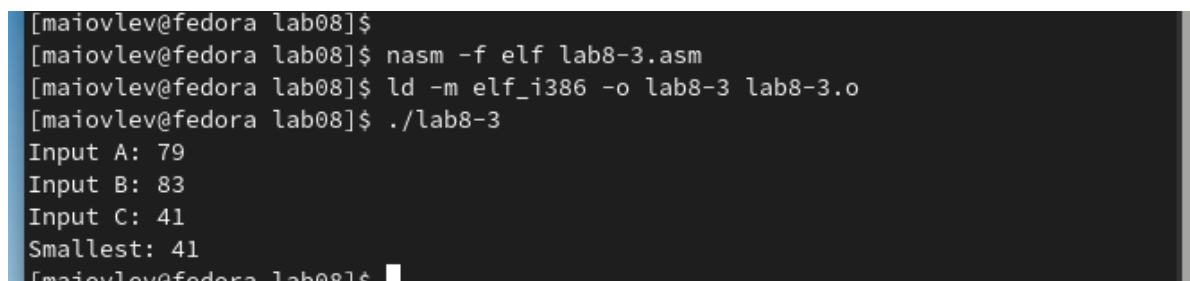
mov eax, msgC
call sprint
mov ecx, C
mov edx, 80
call sread
mov eax, C
call atoi
mov [C], eax...

;-----algorithm-----
....
mov ecx, [A] ;ecx = A
mov [min], ecx ;min = A.

cmp ecx, [B] ; A&B
jl check_C ; if a<b: goto check_C.
mov ecx, [B]
mov [min], ecx ;else min = B

check_C:
cmp ecx, [C]
jl finish
mov ecx, [C]
mov [min], ecx.
```

Рис. 4.12: Файл lab8-3.asm



```
[maiovlev@fedora lab08]$
[maiovlev@fedora lab08]$ nasm -f elf lab8-3.asm
[maiovlev@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[maiovlev@fedora lab08]$ ./lab8-3
Input A: 79
Input B: 83
Input C: 41
Smallest: 41
[maiovlev@fedora lab08]$
```

Рис. 4.13: Программа lab8-3.asm

6. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 8.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений X и a из 8.6. (рис. 4.14, 4.15)

для варианта 6

$$\begin{cases} x + a, x = a \\ 5x, x \neq a \end{cases}$$

```
lab8-4.asm [-M--] 13 L: [ 21+23 44/ 53] *(654 / 741b) 0010 0x00
call atoi.
mov [A],eax

mov eax,msgX
call sprintf
mov ecx,X
mov edx,80
call sread.
mov eax,X
call atoi
mov [X],eax...

;-----algorithm-----

mov ebx, [X]
cmp ebx, [A]
je first
jmp second

first:
mov eax,[X]
mov ebx,[A]
add eax,ebx
call iprintLF.
call quit

second:
mov eax, [X]
mov ebx,5
mul ebx
call iprintLF.
call quit
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню

Рис. 4.14: Файл lab8-4.asm

```
[maiovlev@fedora lab08]$  
[maiovlev@fedora lab08]$ nasm -f elf lab8-4.asm  
[maiovlev@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o  
[maiovlev@fedora lab08]$ ./lab8-4  
Input A: 2  
Input X: 2  
4  
[maiovlev@fedora lab08]$ ./lab8-4  
Input A: 1  
Input X: 2  
10  
[maiovlev@fedora lab08]$
```

Рис. 4.15: Программа lab8-4.asm

5 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.

Список литературы

1. Расширенный ассемблер: NASM
2. MASM, TASM, FASM, NASM под Windows и Linux